Error Pages and Feedback Messages

How do I add custom error pages?

Non-HTTP Error Pages

In your application class that extends WebApplication, set the following in the init() method:

```
getApplicationSettings().setPageExpiredErrorPage(MyExpiredPage.class);
getApplicationSettings().setAccessDeniedPage(MyAccessDeniedPage.class);
getApplicationSettings().setInternalErrorPage(MyInternalErrorPage.class);
```

There are also some other neat settings you should check out in `getApplicationSettings()`, `getDebugSettings()`, `getExceptionSettings()`, `getMarkupSettings()`, `getPageSettings()`, `getRequestCycleSettings()`, `getSecuritySettings` and `getSessionSettings()`. See the javadoc for more information.

If calling `setInternalErrorPage()`, make sure to also call:

```
// show internal error page rather than default developer page
getExceptionSettings().setUnexpectedExceptionDisplay(IExceptionSettings.SHOW_INTERNAL_ERROR_PAGE);
```

You can also add customized error per page. In your WebPage class:

```
getRequestCycle().onRuntimeException(new MyErrorPage(), theException);
```

HTTP Error Pages

First ensure that the wicket filter-mapping in your web.xml has the following dispatchers:

```
	<filter-mapping>
		<filter-name>WicketFilter</filter-name>
		<url-pattern>/*</url-pattern>
		<dispatcher>REQUEST</dispatcher>
		<dispatcher>ERROR</dispatcher>
	</filter-mapping>

	<error-page>
		<error-code>404</error-code>
		<location>/404</location>
	</error-page>
```

This is assuming that you have mapped "/404" in your WebApplication:

```
	mountPage("/404", PageNotFound.class);
```

In your error WebPage:

```
public class PageNotFound extends WebPage {

	private static final long serialVersionUID = 1L;

	public PageNotFound() {
		// Add any necessary components
	}

	@Override
	protected void configureResponse() {
		super.configureResponse();
		getWebRequestCycle().getWebResponse().getHttpServletResponse().setStatus(HttpServletResponse.
SC_NOT_FOUND);
	}

	@Override
	public boolean isVersioned() {
		return false;
	}

	@Override
	public boolean isErrorPage() {
		return true;
	}

}
```

An HTTP error can be thrown manually by:

```
// also see AbortWithWebErrorCodeException and AbortWithHttpStatusException
throw new AbortWithHttpStatusException(404, true);
```

For wicket 6 and beyond:

```
throw new AbortWithHttpErrorCodeException(404, "Some message");
```

Overriding All Error Pages for RuntimeException

Sometimes it is desirable to override all of Wicket's error pages with a custom error page(s) when any RuntimeException occurs. Here is an example of how to do so.

```
public final class MyRequestCycle extends WebRequestCycle {

        /**
         * MyRequestCycle constructor
         *
         * @param application the web application
         * @param request the web request
         * @param response the web response
         */
        public WebRequestCycle(final WebApplication application, final WebRequest request, final Response
response) {
                super(application, request, response);
        }

        /**
         * {@inheritDoc}
         */
        @Override
        protected final Page onRuntimeException(final Page cause, final RuntimeException e) {
                // obviously you can check the instanceof the exception and return the appropriate page if
desired
                return new MyExceptionPage(e);
        }
}
```

Then in your WebApplication class :

```
/**
 * {@inheritDoc}
 */
@Override
public final RequestCycle newRequestCycle(final Request request, final Response response) {
        return new MyRequestCycle (this, (WebRequest)request, (WebResponse)response);
}
```

How do I add/display errors?

Normally all that is needed to display errors in a page is to call the "error" method in the WebPage:

```
error("Message to the user indicating that an error occurred");

// alternative to get the message from a .properties file: error.login.user.invalid=${username} is not a valid
user
error(new StringResourceModel("error.login.user.invalid", this, myUserModelThatHasAUsernameProperty).
getString());
```

If an error occurred where the user needs to be directed to a different page we need to set the error on the session:

```
getSession().error(message);
throw new RestartResponseException(MyErrorPage.class, optionalPageParameters);
// use RestartResponseAtInterceptPageException(MyErrorPage.class);
// instead to interrupt current request processing and immediately redirect to an intercept page

// it may be tempting to do the following, but it should not be done:
// error(message);
// setResponsePage(MyErrorPage.class);
// Why not?

// What about this; will it work? (My experience says no, but I don't know why not.)
// getSession().error(message);
// setRedirect(true);
// setResponsePage(MyErrorPage.class);
```

To display an error (or any message for that matter: info/warning) in an HTML page use a FeedbackPanel:

```
<!-- In the HTML page -->
<div wicket:id="feedback">[Feedback Panel]</div>

// In WebPage
add(new FeedbackPanel("feedback"));
```

How do I add/display errors at the component level?

See ComponentFeedbackPanel, FormComponentFeedbackBorder, and FormComponentFeedbackIndicator

// TODO : add example that adds error messages to individual form components of the page

I get 'Internal error cloning object' errors

This is a debug feature to help find potential problems when the application runs clustered. It checks the component graphs to make sure everything is serializable (which is required for clustering).

If clustering support is not needed, these checks can be disabled by doing `getDebugSettings().setSerializeSessionAttributes(false);` in the `application.init()`

Also, if the `configuration` parameter in the `web.xml` is set to `DEPLOYMENT`, these checks are disabled.

Answer provided courtesy Igor Vaynberg